Docket No. AUS920010225US1 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Hartel et al.**	§	
	§	Group Art Unit: **2174**
Serial No.: **09/882,172**	§	
	§	Examiner: **Ke, Peng**
Filed: **June 14, 2001**	§	
	§	
For: **Property Editor Graphical User Interface Apparatus, Method and Computer Program Product**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300, on August 5, 2005.
By: Amelia C. Turner
Amelia C. Turner

SUPPLEMENTAL APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Request for Reinstatement of Appeal, filed herewith.

The fees required under § 41.20(B)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF REQUEST FOR REINSTATEMENT OF APPEAL.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-37 and 39-41

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 38
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-37 and 39-41
4. Claims allowed: NONE
5. Claims rejected: 1-37 and 39-41
6. Claims objected to: NONE

C. CLAIMS ON APPEAL

The claims on appeal are: 1-37 and 39-41

STATUS OF AMENDMENTS

There are no amendments after final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1:

The presently claimed invention provides a method for editing a property. The present invention identifies one or more methods invoked by a property editor associated with a property. See specification, page 14, lines 1-16; page 15, lines 9-15; page 17, lines 4-10 and 16-27; page 19, lines 1-7; page 20, lines 3-8; page 23, lines 16-18. The present invention selects a graphical user interface based on the one or more methods invoked by the property editor and the present invention presents the graphical user interface to use in editing the property. See specification, page 15, lines 12-19; page 16, lines 4-10; page 17, lines 11-15; page 21, lines 18-24. Examples of properties, property editors, and graphical user interfaces are provided in the specification on page 14, line 16, to page 16, line 12; page 18, lines 6-13; **Figures 4A-4C, 5A-5C, 6A, and 6B.**

Independent claim 13:

The presently claimed invention provides an apparatus for editing a property. The present invention identifies one or more methods invoked by a property editor associated with a property. See specification, page 14, lines 1-16; page 15, lines 9-15; page 17, lines 4-10 and 16-27; page 19, lines 1-7; page 20, lines 3-8; page 23, lines 16-18. The present invention selects a graphical user interface based on the one or more methods invoked by the property editor and the present invention presents the graphical user interface to use in editing the property. See specification, page 15, lines 12-19; page 16, lines 4-10; page 17, lines 11-15; page 21, lines 18-24. Examples of properties, property editors, and graphical user interfaces are provided in the specification on page 14, line 16, to page 16, line 12; page 18, lines 6-13; **Figures 4A-4C, 5A-5C, 6A, and 6B.**

The means recited in independent claim 13, as well as dependent claims 14-24, may be data processing hardware within server 104 or one of clients 108, 110, 112 operating under control of software performing the steps described in the specification at page 21, lines 11-24, or equivalent.

Independent claim 25:

The presently claimed invention provides a computer program product for editing a property. The present invention identifies one or more methods invoked by a property editor associated with a property. See specification, page 14, lines 1-16; page 15, lines 9-15; page 17, lines 4-10 and 16-27; page 19, lines 1-7; page 20, lines 3-8; page 23, lines 16-18. The present invention selects a graphical user interface based on the one or more methods invoked by the property editor and the present invention presents the graphical user interface to use in editing the property. See specification, page 15, lines 12-19; page 16, lines 4-10; page 17, lines 11-15; page 21, lines 18-24. Examples of properties, property editors, and graphical user interfaces are provided in the specification on page 14, line 16, to page 16, line 12; page 18, lines 6-13; **Figures 4A-4C, 5A-5C, 6A, and 6B.**

A person having ordinary skill in the art would be able to derive computer instructions on a computer readable medium given **Figure 7** and the corresponding description at page 21, lines 11-24, or the pseudo-code presented on pages 22-28, without undue experimentation.

Independent claim 37:

In addition to the above, the present invention may also identify one or more methods invoked by a property editor associated with a property, wherein the one or more methods include one or more PropertyEditor Interface methods and wherein the property is a data type. See specification, page 15, line 8, to page 16, line 10; page 19, lines 1-7; page 20, lines 3-8. The present invention selects a graphical user interface based on the one or more methods invoked by the property editor and the present invention presents the graphical user interface to use in editing the property. See specification, page 21, lines 18-24. Examples of methods in the PropertyEditor Interface are provided in the **Table 1**.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection on appeal are as follows:

I. Claims 1-3, 7, 8, 13-15, 19, 20, 25-27, 31, 32, and 37 are rejected under 35 U.S.C. § 102(e) as being allegedly anticipated by *Miller et al.* (U.S. Patent No. 6,661,437);

II. Claims 4-6, 16-18, and 28-30 are rejected under 35 U.S.C. § 103(a) as being allegedly obvious over *Miller et al.* (U.S. Patent No. 6,661,437) and further in view of *Lindhorst et al.* (U.S. Patent No. 6,337,696);

III. Claims 9, 12, 21, 24, 33, 36, 40, and 41 are rejected under 35 U.S.C. § 103(a) as being allegedly obvious over *Miller et al.* (U.S. Patent No. 6,661,437) in view of *Zimmerman et al.* (U.S. Patent No. 6,417,872);

IV. Claims 10, 11, 22, 23, 34, 35, and 39 are rejected under 35 U.S.C. § 103(a) as being allegedly obvious over *Miller et al.* (U.S. Patent No. 6,661,437) in view of *Zimmerman et al.* (U.S. Patent No. 6,417,872) and further in view of *Lindhorst et al.* (U.S. Patent No. 6,337,696).

ARGUMENT

I. 35 U.S.C. § 103, Alleged Anticipation of Claims 1-3, 7, 8, 13-15, 19, 20, 25-27, 31, 32, and 37

The Final Office Action rejects claims 1-3, 7, 8, 13-15, 19, 20, 25-27, 31, 32, and 37 under 35 U.S.C. § 102(e) as being allegedly obvious over *Miller et al.* (U.S. Patent No. 6,661,437). This rejection is respectfully traversed.

With reference to claim 1, the Final Office Action states:

> As per claim 1, Miller teaches a method in a data processing system, for editing a property, comprising:
>
> identifying one or more methods invoked by a property editor associated with the property; (See Miller figure 2; items 225, 230, 235, 240; Examiner interprets each item under setup to be a method associated with the property of setup)
>
> selecting a graphical user interface based on the one or more methods invoked by the property editor; (See Miller figure 4, item 415, figure 5, item 545; Since GUI invoked for subtitle language is different from that of the Subtitle display therefore property editor selected a GUI based on the method) and
>
> providing the graphical user interface for use in editing the property (figure 5, item 545).

Final Office Action dated April 8, 2005. *Miller* teaches an on-screen user interface display that generates a display of multiple hierarchically ordered menus. See *Miller*, col. 1, line 31-40. More particularly, *Miller* is concerned with graphical user interfaces for interactive operation of consumer electronics devices, such as a DVD player. See *Miller*, col. 2, line 23-29.

FIG. 2 of *Miller*, which is an example of an on-screen user interface display indicating a base menu and a navigation path through hierarchical DVD operational menus, is as follows:

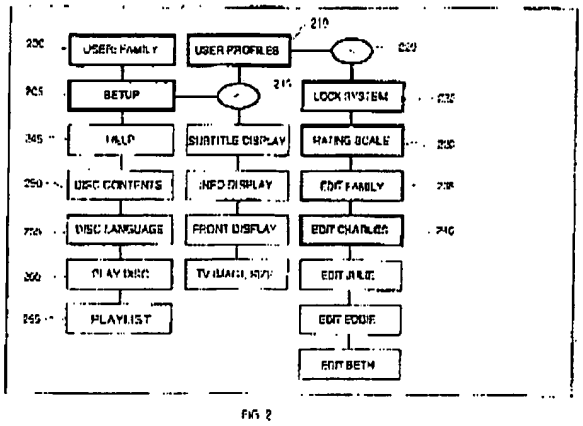


FIG. 2

With respect to **FIG. 2**, *Miller* states:

> FIG. 2 shows an on-screen User interface display indicating a base menu and a navigation path through three hierarchical DVD operational menus. Specifically, FIG. 2 identifies the navigation path that a User has followed from a base menu item 205, through entry item 215 and item 210 of a second menu and through menu items 220-235 to item 240 of a third and currently active menu. The base menu in this example comprises option items 200, 205, 245, 250, 255, 260, and 265. The navigation path through items 200-240 is identified to the User by bold highlighting and is traversed following User entry of directional or cursor commands on remote unit 70 (FIG. 1). The menu entry points are indicated by items 215 and 220 (FIG. 2). The menu exit points are identified by a highlighted navigation path from the preceding menu (e.g. from item 210) to the entry point of the next menu (e.g. item 220). The entry and exit points and any intervening items of an individual menu are identified by bold highlighting. This enables a User to readily see the menu navigation path and operation sequence he has followed. In another embodiment, the entry and exit points of a particular menu may be identified (e.g. by highlighting) and intervening items may not be highlighted for identification.

Miller, col. 6, lines 25-47. Thus, items **200, 205, 245, 250, 255, 260,** and **265** comprise a base menu. When a user selects item **205**, for example, the items "USER PROFILES," "SUBTITLE DISPLAY," "INFO DISPLAY," "FRONT DISPLAY," and "TV IMAGE SIZE" are displayed. Then, when the user selects item **210**, "USER PROFILES," items **225-240** are displayed. In other words, the menu system of *Miller* selects user interface components to display based on items selected in a menu.

First, the Final Office Action states that the Examiner interprets items under setup to be a method invoked by a property editor associated with a property. The remainder of the rejection relies on this interpretation of a "method" being an item under setup in a user interface. For example, the Final Office Action concludes that *Miller* teaches selecting a graphical user interface based on the one or more methods invoked by the property editor. This conclusion can only be made based on the assumption that *Miller* actually teaches identifying the one or more methods invoked by the property editor. This initial interpretation is not reasonable and does not address the specific limitations presented in the instant claims for the reasons presented below.

Claim 1, which is representative of claims 13 and 25 with regard to similarly recited subject matter, reads as follows:

> 1. A method, in a data processing system, for editing a property, comprising:
> identifying one or more **methods invoked by a property editor** associated with the property;
> selecting a graphical user interface **based on the one or more methods invoked by the property editor**; and
> providing the graphical user interface for use in editing the property.
> [emphasis added]

Thus, claim 1, for example, clearly recites identifying one or more methods **invoked** by a **property editor** associated with a property. *Miller* simply teaches a hierarchical set of menus. *Miller* does not contemplate identifying one or more **methods invoked** by a property editor. For example, one property that may be interpreted to be edited in *Miller* is the "subtitle language" operational parameter. **FIG. 5** of *Miller* presents a user interface for editing the subtitle language parameter, as reproduced below:

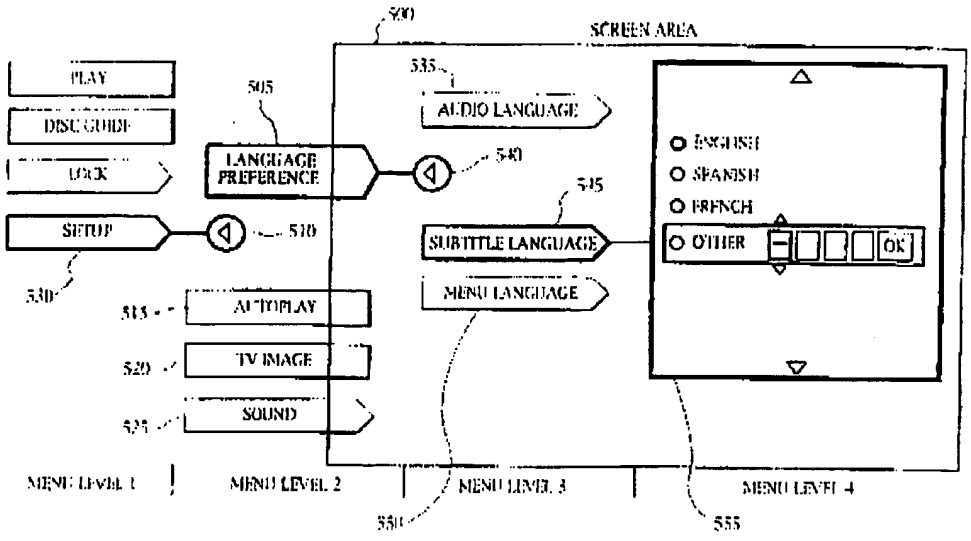


FIG. 5

Regarding **FIG. 5**, *Miller* states:

> FIG. 5 shows an on-screen User interface display, according to the invention, depicting automatic scrolling operation for maintaining User viewability of the currently selectable menu items in a hierarchical menu structure. In this example, a User has progressed through the menu structure from menu level 1 to the currently active menu level 4 via highlighted items 530, 510, 505, 540, 545 and 555. Controller 60 operating in conjunction with unit 37 (FIG. 1) has automatically scrolled the displayed portion of the menu structure horizontally to position the dialog box 555 (menu level 4) at the right of the displayed screen and preceding menu (menu level 3) within screen area 500. As a result, the previously selected menus (levels 1 and 2) have been scrolled from right to left and off-screen leaving displayed screen image 500. The displayed menu structure portion includes dialog box 555 (menu level 4) permitting a User to select a desired subtitle language. A User is able to readily identify and re-trace his highlighted navigation path and may advance through the menu structure knowing that the menus automatically scroll to keep the next menu selection visible and on-screen for the duration of their use.

Miller, col. 7, lines 44-60. With respect to how the user interface display of **FIG. 5**, for example, is generated, *Miller* states that the on-screen display and graphics generator creates the interactive menu displays. This is done in *Miller* using a database that links particular menus in the hierarchical structure with selectable menu option icons. The controller determines and

generates the next hierarchical menu to be displayed in response to user selection of a current menu option icon. See *Miller*, col. 4, lines 20-37. In other words, the menus and the menu option icons are predetermined as defined by the database.

Similarly, the user interface components for a parameter are static. *Miller* does not teach or suggest **selecting** a graphical user interface based on the one or more methods invoked by a property editor associated with a property. That is, the slide bar for adjusting the brightness parameter in **FIG. 9** will always be a slide bar. This same graphical user interface will always be provided in *Miller* for the brightness parameter when the user selects the appropriate menu item. There is no identification of one or more methods invoked by a property editor in *Miller* for selecting a graphical user interface for the property editor.

Furthermore, the broadest reasonable interpretation of the claims must be consistent with the interpretation that those skilled in the art would reach. *In re Cortright*, 165 F.3d 1353, 1359; 49 USPQ2d 1464, 1468 (Fed. Cir. 1999). Also, the claims must be "given [their] broadest reasonable interpretation consistent with the specification." *In re Hyatt*, 211 F.3d 1367, 1372; 54 USPQ2d 1664, 1667 (Fed. Cir. 2000). In this case, when the claims are read in light of the specification, it is clear that the term "method" is used in the context of an object oriented runtime environment. More specifically, the present specification states:

> When an application is executed on a JVM that is implemented in software on a platform-specific operating system, a Java application may interact with the host operating system by invoking native methods. A Java method is written in the Java language, compiled to bytecodes, and stored in class files. A native method is written in some other language and compiled to the native machine code of a particular processor. Native methods are stored in a dynamically linked library whose exact form is platform specific.
>
> As mentioned above, the present invention is directed to a mechanism for determine the type of graphical user interface (GUI) to be used with a Java property editor to edit a Java property. A "property" as the term is used herein, refers to anything that can be represented by a Java class, i.e. a data type. Examples of properties include text color, background color, text string, windspeed, percentage, computer operating system, and the like.
>
> Each property has an associated property editor defined by a programmer using one or more methods for displaying and updating property values. These methods may be custom methods created by the programmer or may be methods defined in the Java PropertyEditor interface. The Java PropertyEditor interface defines a plurality of methods that may be used and combined to generate property editors.

See specification, page 13, line 21, to page 14, line 15. It is clear to a person of ordinary skill in the art that the terms "method," "property," and "property editor" would be given their customary meanings **within the relevant art**. Claim terms are presumed to have the ordinary and customary meanings attributed to them by those of ordinary skill in the art. *Sunrace Roots Enter. Co. v. SRAM Corp.*, 336 F.3d 1298, 1302; 67 USPQ2d 1438, 1441 (Fed. Cir. 2003); *Brookhill-Wilk 1, LLC v. Intuitive Surgical, Inc.*, 334 F.3d 1294, 1298; 67 USPQ 2d 1132, 1136 (Fed. Cir. 2003). It is the use of the words in the context of the written description and customarily by those of ordinary skill in the relevant art that accurately reflects both the "ordinary" and the "customary" meaning of the terms in the claims. *Ferguson Beauregard/Logic Controls v. Mega Systems*, 350 F.3d 1327, 1338; 69 USPQ2d 1001, 1009 (Fed. Cir. 2003).

Again, in the present case, the phrase, "one or more methods invoked by a property editor associated with the property," as recited in claim 1 for instance, has a more specific meaning to a person of ordinary skill in the relevant art. Here, the relevant art concerns object-oriented runtime environments, such as a Java runtime environment. In such a runtime environment, as recited in claim 1, for example, a property editor associated with a property to be edited **invokes** methods for displaying and updating property values. *Miller* makes no mention of a property editor that invokes methods, particularly in an object-oriented runtime environment. In fact, *Miller* does not include a single occurrence of the word "method" in this context. Thus, *Miller* fails to teach or fairly suggest the steps of "identifying one or more methods invoked by a property editor associated with the property" and "selecting a graphical user interface based on the one or more methods invoked by the property editor" either on its face or when interpreted within the context of an object-oriented runtime environment.

For the above reasons, *Miller* does not anticipate claim 1. Independent claims 13 and 25 recite subject matter addressed above with respect to claim 1 and are allowable for similar reasons. Since claims 2, 3, 7, 8, 14, 15, 19, 20, 26, 27-12, 14-24, 26-36, and 39-41 depend from claims 1, 13, 25, and 37, the same distinctions between *Zimmerman* and *Carter* and the invention recited in claims 1, 13, 25, and 37 apply for these claims. Additionally, claims 2-12, 14-24, 26-36, and 39-41 recite additional features not suggested by the references.

Accordingly, Appellants respectfully request that the rejection of dependent claims 1-3, 7-9, 12-15, 19-21, 24-27, 31-33, 36, 37, 40, and 41 under 35 U.S.C. § 102(e) not be sustained.

I.A. 35 U.S.C. § 103, Alleged Anticipation of Claims 3, 15, 27, and 37

With regard to claim 3, the Final Office Action alleges that *Miller* teaches that one or more methods invoked by the editor include one or more PropertyEditor Interface methods in **FIG. 4**, item **415**, and **FIG. 5**, item **545**. The Final Office Action apparently equates a graphical user interface icon or menu item as a PropertyEditor Interface method. As stated above, the claim language must be given an interpretation that is consistent with the interpretation that those skilled in the art would reach. The present specification defines the PropertyEditor Interface as follows:

> The Java PropertyEditor interface defines a plurality of methods that may be used and combined to generate property editors.

The Final Office Action makes no effort whatsoever to explain why an item in a menu is somehow equivalent to the claimed PropertyEditor Interface methods.

Appellants assert that the applied reference fails to teach or suggest each and every claim limitation. The Final Office Action has failed to point out where each claim limitation is taught or suggested in the applied prior art other than to simply refer to a menu item, whether the menu item is equivalent or not. Therefore, the Final Office Action fails to establish a *prima facie* case of anticipation for claims 3, 15, 27, and 37.

I.B. 35 U.S.C. § 103, Alleged Anticipation of Claims 7, 8, 19, 20, 31, and 32

With regard to claims 7, 8, 19, 20, 31 and 32, *Miller* fails to teach that if one or more abilities of the property editor include an ability to edit a property using tags, the graphical user interface includes at least one of a popup choice selection area virtual button and a current selection display field. The Office Action alleges that this feature is taught at col. 7, lines 1-36, which reads:

> In this example, a User has progressed through the menu structure from menu level 1 to the currently active menu level 4 via highlighted items 530, 510, 505, 540, 545 and 555. Controller 60 operating in conjunction with unit 37 (FIG. 1) has automatically scrolled the displayed portion of the menu structure horizontally to position the dialog box 555 (menu level 4) at the right of the displayed screen and preceding menu (menu level 3) within screen area 500. As a result, the previously selected menus (levels 1 and 2) have been scrolled from right to left and off-screen leaving displayed screen image 500. The displayed menu structure portion includes dialog box 555 (menu level 4) permitting a User to select a desired subtitle language. A User is able to readily identify and re-trace his highlighted navigation path and may advance through the menu structure knowing that the menus automatically scroll to keep the next menu selection visible and on-screen for the duration of their use.

While this section of *Miller* teaches the use of a dialog box **555**, there is nothing in this, or any other, section of *Miller* that teaches, or even suggests, that this dialog box **555** is provided when the one or more methods invoked by a property editor associated with a property to be edited identify an ability of the property editor as being able to edit a property using tags. Thus, while *Miller* may teach a dialog box, *Miller* still does not teach or suggest the specific features of claims 7, 8, 19, 20, 31, and 32.

II. 35 U.S.C. § 103, Alleged Obviousness of Claims 4-6, 16-18, and 28-30

The Office Action rejects claims 4-6, 16-18, and 28-30 under 35 U.S.C. § 103(a) as being allegedly unpatentable over *Miller* (U.S. Patent No. 6,661,437) in view of *Lindhorst et al.* (U.S. Patent No. 6,337,696). This rejection is respectfully traversed for at least the same reasons as noted above with regard to claims 1, 13, and 25 from which claims 4-6, 16-18, and 28-30 depend.

Specifically, *Miller* fails to teach identifying one or more methods invoked by a property editor for the property that is to be edited. Similarly, *Miller* does not teach selecting a graphical user interface based on the one or more methods invoked by the property editor. In addition, *Lindhorst* does not provide for the deficiencies of *Miller*. That is, neither *Miller* nor *Lindhorst*, taken alone or in combination, teaches or suggests identifying one or more methods invoked a property editor for a property that is to be edited or selecting a graphical user interface based on the one or more methods invoked by the property editor.

Lindhorst is directed to a method for creating and editing event handlers that link events triggered on one object to take actions on one or more different objects. A graphical user interface contains three different panes, an event pane, an action pane and a code pane. A user selects an event icon in an event pane to link that event to a desired action in the action pane. The generated code can then be viewed in the code pane.

In column 18, lines 18-68, *Lindhorst* discusses editing of properties using one of a "String Dialog Box," a "Color Dialog Box," and a "Number Dialog Box" based on the type of property determined at state **410** in **Figure 6** of *Lindhorst*. Thus, while *Lindhorst* discloses a different editor interface for editing the value of a property based on the identified property type, *Lindhorst* still does not teach or suggest identifying one or more methods invoked by a property editor associated with a property to be edited and then selecting a graphical user interface based on the one or more methods invoked by the property editor. To the contrary, the *Lindhorst* reference merely identifies a type of the property and determines a dialog box to be displayed based on the type of property. Thus, any alleged combination of *Lindhorst* with *Miller*, even if such a combination were somehow possible and one were motivated to attempt it, still would not result in the invention recited in independent claims 1, 13, and 25.

Furthermore, there is no motivation in the prior art for combining *Miller* and *Lindhorst*. The Examiner may not use the claimed invention as an "instruction manual" or "template" to piece together the teachings of the prior art so that the invention is rendered obvious. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Such reliance is an impermissible use of hindsight with the benefit of applicant's disclosure. *Id.* Therefore, absent some teaching, suggestion, or incentive in the prior art, *Miller* and *Lindhorst* cannot be properly combined to form the claimed invention. As a result, absent any teaching, suggestion, or incentive from the

prior art to make the proposed combination, the presently claimed invention can be reached only through an impermissible use of hindsight with the benefit of applicant's disclosure a model for the needed changes. In this case, *Miller* does not recognize a problem for which a system and method for facilitating generation and editing of event handlers, as taught by *Lindhorst*, is a solution. In actuality, *Miller* makes not mention of an object-oriented runtime environment or editing of event handlers. Therefore, a person of ordinary skill in the art would not look to *Lindhorst* as a reasonable solution to any problems recognized in *Miller*.

In view of the above, Appellants respectfully submit that neither *Miller* nor *Lindhorst*, taken alone or in combination, teaches or suggests each and every feature of independent claims 1, 13, and 25. Since claims 4-6, 16-18, and 28-30 depend on claims 1, 13, and 25, the same distinctions between *Miller* and *Lindhorst* apply for these claims. Accordingly, Appellants respectfully request that the rejection of dependent claims 4-6, 16-18, and 28-30 under 35 U.S.C. § 103(a) not be sustained.

III. 35 U.S.C. § 103, Alleged Obviousness of Claims 9, 12, 21, 24, 33, 36, 40, and 41

The Final Office Action rejects claims 9, 12, 21, 24, 33, 36, 40, and 41 under 35 U.S.C. § 103(a) as being allegedly unpatentable over *Miller et al.* (U.S. Patent No. 6,661,437) in view of *Zimmerman et al.* (U.S. Patent No. 6,417,872). This rejection is respectfully traversed for at least the same reasons as noted above with regard to claims 1, 13, 25, and 37 from which claims 9, 12, 21, 24, 33, 36, 40, and 41 depend.

Specifically, *Miller* fails to teach identifying one or more methods invoked by a property editor for the property that is to be edited. Similarly, *Miller* does not teach selecting a graphical user interface based on the one or more methods invoked by the property editor. In addition, *Zimmerman* does not provide for the deficiencies of *Miller*. That is, neither *Miller* nor *Zimmerman*, taken alone or in combination, teaches or suggests identifying one or more methods invoked a property editor for a property that is to be edited or selecting a graphical user interface based on the one or more methods invoked by the property editor.

Zimmerman is directed to a system for adding application-defined properties and application-defined property sheet pages to a list of system defined properties. Once added, the application-defined properties may be displayed and edited. In addition, a user may select

several objects, display the properties common to all of the objects in a list, and edit the common properties. Further, a user may select several objects, display the property sheet pages common to all of the objects, and edit the properties on these property sheet pages. Also, a user may switch between viewing a property in a list of properties and viewing a property on a property sheet page.

Thus, *Zimmerman* is concerned with three main objectives. The first objective is providing a system that allows a user to view and edit application-defined properties as well as system-defined properties. The second objective *Zimmerman* is concerned with is allowing a user to view and edit several objects, having common properties, simultaneously. Finally, the third objective of *Zimmerman* is to provide a system that allows a user to switch between per-property browsing and property sheet page methods. While *Zimmerman* may speak generally of editing object properties, *Zimmerman* does not teach identifying one or more methods associated with a property editor. Similarly, *Zimmerman* does not teach selecting a graphical user interface based on the one or more methods associated with the property editor.

In other words, claims 1, 13, 25, and 37 recite selecting a graphical user interface, to edit properties, based on the methods invoked by a particular property editor for the property. In *Zimmerman*, the graphical user interface is selected by the user when navigating between a per-property browser and a property sheet page (column 7, lines 49-63). This feature is described in more detail at column 8, line 35 - column 9, lines 22, of *Zimmerman*, which reads as follows:

> Fig. 9 is a flowchart of the **steps performed by the per-property browser when either the drop down list button or the property sheet page map button have been activated.** First, the per-property browser receives user input which indicates which step the per-property browser should take next (step 80). **Then the per-property browser determines whether the user input is a request to activate a drop down list button, which indicates a desire to view the drop down list (step 82).** If the user input requests activation of the drop down list button, then the per-property browser calls the GetPredefinedStrings ( ) function to obtain character strings corresponding to the possible values of the property (step 84). Then the drop down list is displayed (step 86). If the user input is not a request to activate the drop down list button, then the per-property browser determines if the user input is a request to select an element in a drop down list which is already displayed (step 88). If the user input is a request to select an element, then the per-property browser calls the

> GetPredefinedValue ( ) function to obtain the value corresponding to the user's choice (step 90). Then this value replaces the current property value (step 91). For example, if the per-property browser receives user input requesting activation of the drop down list button for a color property, which allows selection of a color for text, then the per-property browser calls the GetPredefinedStrings ( ) function. The GetPredefinedStrings ( ) function may return character strings such as "Red," "Blue," and "Green," which corresponds to possible values of the color property. When the user selects one of these character strings, such as "Red," the per-property browser calls the GetPredefinedValue ( ) function. The GetPredefinedValue ( ) function returns the actual value corresponding to the selected character string, such as an RGB value for the selected color "red."
>
> Continuing with the flowchart, if the user input is not a request to select an element, then the per-property browser determines if the user input is a request to activate the property sheet page map button (step 92). **If the user input is a request to activate a property sheet page map button, then the user desires to view the currently selected property on a property sheet page.** The per-property browser calls the MapPropertyToPage ( ) function to display the currently selected property on a property sheet page (step 93). In particular, the per-property browser calls the MapPropertyToPage ( ) function to obtain an application defined property sheet page identifier for a property sheet page. Then, the per-property browser invokes the property sheet page browser, passing to it the class identifier returned by the MapPropertyToPage ( ) function. [emphasis added]

Thus, in *Zimmerman*, the user can view a graphical user interface containing a drop down list. In addition, the user can view a property sheet page graphical user interface by activating a property sheet page map button. Although the user may select the property sheet page graphical user interface by activating a property sheet page map button, the graphical user interface is selected to switch between a property sheet page graphical user interface and a graphical user interface containing a drop down list of properties. In other words, the graphical user interface is not selected based on the methods invoked by a property editor associated with a property that is to be edited but rather, on the selections made by the user.

The present invention permits a property and an associated property editor to be defined. Different graphical user interfaces may be provided based on the different methods that may be invoked by property editors. The present invention determines what methods are invoked by the

property editor for the property that is to be edited and then, based on the identification of these methods, a graphical user interface may be selected for graphically representing the property editor. *Zimmerman* does not provide such functionality. To the contrary, in *Zimmerman*, regardless of the type of property or property editor that may be associated with the property, a user may select either the property sheet page graphical user interface or the drop down list of properties graphical user interface. The graphical user interface that is selected has no relation to the property editors for the properties or the methods invoked by the property editors of the properties.

IIIA. 35 U.S.C. § 103, Alleged Obviousness of Claims 9, 12, 21, 24, 33, and 36

More particularly, with regard to claims 9, 21 and 33, *Zimmerman* does not teach that if the one or more abilities include an ability to edit the property using a custom editor interface, the graphical user interface includes a popup custom component area virtual button. The Office Action alleges this feature is taught at column 6, lines 53-64, of *Zimmerman*, which reads as follows:

> A property sheet page browser 42 is within a window 40. The property sheet page browser 42 is part of the development environment, and it provides the framework within which an application defined property sheet page 44 displays itself. The property sheet page browser 42 contains a tab 45 which identifies a property group. For instance, tab 45 indicates that the property sheet page is the "Font" property group. Other tabs 46 are also displayed alongside tab 45 in such a way that the display resembles a set of tabbed index cards. The other tabs 46 represent other property groups associated with a particular object. The application defined property sheet page 44 contains various properties of an object which may be displayed and edited.

This section teaches selecting a tab in a property sheet browser to view property groups associated with an object. The Examiner interprets the tabs representing the property groups in *Zimmerman* to be a popup custom component area virtual button because by selecting a tab, a custom property sheet page is allegedly displayed (Final Office Action, page 6). While a property sheet page containing application-defined properties may be displayed, this section does not teach a custom editor. The present specification provides an exemplary definition of a custom editor as an editor that uses methods created by the programmer rather than defined by

the programming language (see specification, page 14, lines 9-16). A custom property editor is simply not taught in *Zimmerman*. Nowhere is anything mentioned in *Zimmerman* regarding a programmer defining methods to edit properties. Thus, as *Zimmerman* does not teach a custom editor, *Zimmerman* cannot teach a custom editor interface.

Furthermore, there is no teaching in *Zimmerman* as to the identification of one or more methods invoked by a property editor that identifying abilities of the property editor as including the ability of editing a property using a customer editor interface and, based on this identification, providing a graphical user interface that includes a popup custom component area virtual button, as recited in claims 9, 21 and 33. Thus, despite the allegations made by the Office Action, the *Zimmerman* reference, in actuality, does not teach or even suggest the specific features recited in claims 9, 21 and 33.

Therefore, even if a one were motivated to combine *Miller* and *Zimmerman*, the proposed combination would not result in the claimed invention. That is, *Miller* merely teaches a hierarchical menu for an electronics device, such as a DVD player. *Zimmerman* merely teaches that a graphical user interface is not selected based on the methods invoked by a property editor associated with a property that is to be edited but, rather, on the selections made by the user. The applied references, taken alone or in combination, simply fail to teach or fairly suggest the limitations recited in claim 9, for example. Claims 21 and 33 recite subject matter addressed above with respect to claim 9 and are allowable for similar reasons. Claims 12, 24, and 36 depend from claims 9, 21, and 33, respectively, and are allowable at least by virtue of their dependency.

Furthermore, non-analogous art cannot be used to establish obviousness. *In re Pagliaro*, 210 U.S.P.Q. 888, 892 (C.C.P.A. 1981). Although one of ordinary skill in the art is presumed to be aware of all prior art in the field to which the claimed invention pertains, he or she is not presumed to be aware of prior art outside of that field and the field of the problem to be solved. The presently claimed invention is directed towards the field of providing graphical user interfaces for property editors, while *Miller* is not directed towards property editors in an object-oriented runtime environment at all. Therefore, *Miller* is non-analogous art and cannot be used to form a *prima facie* case of obviousness.

Moreover, *Miller* does not recognize a problem for which the teachings of *Zimmerman* would be a solution. Therefore, absent some teaching, suggestion, or incentive in the prior art, *Miller* and *Zimmerman* cannot be properly combined to form the claimed invention. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed combination, the presently claimed invention can be reached only through the impermissible use of hindsight with the benefit of Appellants' disclosure as a model for the needed changes.

III.B. 35 U.S.C. § 103, Alleged Obviousness of Claim 40

Regarding claim 40, *Zimmerman* does not teach that the one or more methods include a getTags method. The Final Office Action alleges the "GetPredefinedStrings" function in *Zimmerman* is synonymous to the getTags method in the present invention. Appellants respectfully disagree. For the same reasons as noted above, *Zimmerman* does not teach using a custom editor. In addition, the "GetPredefinedStrings" function returns character strings that correspond to possible values of the color property (*Zimmerman*, column 8, lines 60-67). This has nothing to do with supporting a custom editor as does the getTags method of the present invention. As described in Table 1 of the present specification, getTags() is a method that returns an array of tags if the property value must be one of a set of known tagged values. This function is not performed by the "GetPredefinedStrings" function of *Zimmerman*.

III.C. 35 U.S.C. § 103, Alleged Obviousness of Claim 41

Regarding claim 41, *Zimmerman* does not teach that the one or more methods include at least one of a supportsCustomEditor method and a getCustomEditor method. The Office Action alleges the "MapPropertyToPage()" function in *Zimmerman* is synonymous to the getCustomEditor method in the present invention. Appellants respectfully disagree. For the same reasons as noted above, *Zimmerman* does not teach using a custom editor. In addition, the "MapPropertyToPage()" function enables switching from a per-property browsing list to a property sheet page containing the particular property (*Zimmerman*, column 8, lines 20-23). This has nothing to do with supporting a custom editor as does the supportsCustomEditor method and the getCustomEditor method of the present invention. As described in Table 1 of the present specification, getCustomEditor() is a method that permits a full customer component

that edits the property to be made available and the supportsCustomEditor method determines whether the property editor supports a custom editor. Neither of these functions is performed by the "MapPropertyToPage()" function of *Zimmerman*.

IV. 35 U.S.C. § 103, Alleged Obviousness of Claims 10, 11, 22, 23, 34, 35, and 39

The Final Office Action rejects claims 10, 11, 22, 23, 34, 35, and 39 under 35 U.S.C. § 103(a) as being allegedly obvious over *Miller et al.* (U.S. Patent No. 6,661,437) in view of *Zimmerman et al.* (U.S. Patent No. 6,417,872) and further in view of *Lindhorst et al.* (U.S. Patent No. 6,337,696). This rejection is respectfully traversed for at least the same reasons as noted above with regard to claims 1, 13, 25, and 37 from which claims 10, 11, 22, 23, 34, 35, and 39 depend. Claim 39 appears to be included in this ground of rejection erroneously, as the body of the rejection addresses claim 39 with respect to *Miller* and *Zimmerman* only. Claim 39 is believed to be allowably by virtue of its dependency on claim 37, which is addressed above.

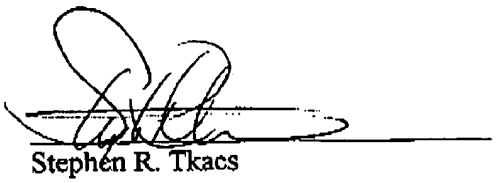
In addition to the above, *Miller*, *Zimmerman*, and *Lindhorst*, taken alone in combination, fail to teach or suggest the specific features of claims 10, 11, 22, 23, 34, and 35. Specifically, neither *Miller*, *Zimmerman*, nor *Lindhorst* teaches or suggests an ability to edit a property using a custom editor interface. This feature is not taught by *Zimmerman* for the same reasons as noted above. Further, *Lindhorst* does not provide for the deficiencies of *Miller* and *Zimmerman*. That is, *Lindhorst* teaches nothing about custom editing as defined in the present invention. To the contrary, *Lindhorst* uses software components that are pre-existing modules, i.e. the String, Color, and Number Dialog Boxes that do not need to be modified to work within embodiments of the *Lindhorst* invention. There is nothing in *Lindhorst* that has anything to do with creating custom components or custom editors. *Lindhorst* merely uses methods defined by pre-existing components. Thus, *Miller*, *Zimmerman*, and *Lindhorst*, taken individually or in combination, fail to teach or suggest an ability to edit a property using a custom editor interface.

Since the applied references fail to teach or suggest the use of a custom editor interface, the references also fail to teach or suggest that when a property editor is determined to use a customer editor interface, a graphical user interface includes a popup custom component area virtual button and at least one of a text entry field and an entry error indicator (claims 10, 22, 34). Similarly, the applied references fail to teach or suggest that such an entry error indicator is

only displayed when an invalid entry in the text field entry area of the graphical user interface is generated based on the fact that the property editor uses a custom editor interface (claims 11, 23, 35). Thus, in addition to being allowable by virtue of their dependency, claims 10, 11, 22, 23, 34, 35, and 39 are also allowable over the proposed combination of references by virtue of the specific features recited in these claims.

CONCLUSION

In view of the above, Appellants respectfully submit that claims 1-37 and 39-41 are allowable over the cited prior art and that the application is in condition for allowance. Accordingly, Appellants respectfully request the Board of Patent Appeals and Interferences to not sustain the rejections set forth in the Final Office Action.



Stephen R. Tkacs
Reg. No. 46,430
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal reads:

1. A method, in a data processing system, for editing a property, comprising:

identifying one or more methods invoked by a property editor associated with the property;

selecting a graphical user interface based on the one or more methods invoked by the property editor; and

providing the graphical user interface for use in editing the property.

2. The method of claim 1, wherein the one or more methods invoked by the property editor identify one or more abilities of the property editor.

3. The method of claim 1, wherein the one or more methods invoked by the editor include one or more PropertyEditor Interface methods.

4. The method of claim 2, wherein if the one or more abilities include a text editing ability, the graphical user interface includes a text field entry area.

5. The method of claim 4, wherein if the one or more abilities include a text editing ability, the graphical user interface further includes an entry error indicator.

6. The method of claim 5, wherein the entry error indicator is only visible when an entry in the text field entry area is invalid.

7. The method of claim 2, wherein if the one or more abilities include an ability to edit a property using tags, the graphical user interface includes at least one of a popup choice selection area virtual button and a current selection display field.

8. The method of claim 7, wherein if the popup choice selection area virtual button is selected, a choice selection area popup is presented.

9. The method of claim 2, wherein if the one or more abilities includes an ability to edit the property using a custom editor interface, the graphical user interface includes a popup custom component area virtual button.

10. The method of claim 9, wherein if the one or more abilities includes an ability to edit the property using a custom editor interface, the graphical user interface further includes at least one of a text entry field and an entry error indicator.

11. The method of claim 10, wherein the entry error indicator is only displayed when an invalid entry is entered in the text field entry area.

12. The method of claim 9, wherein a custom component area is presented in response to selection of the popup custom component area virtual button, and wherein the custom component area includes a custom editor for the property.

13. An apparatus for editing a property, comprising:

means for identifying one or more methods invoked by a property editor associated with the property;

means for selecting a graphical user interface based on the one or more methods invoked by the property editor; and

means for providing the graphical user interface for use in editing the property.

14. The apparatus of claim 13, wherein the means for identifying the one or more methods invoked by the property editor identifies one or more abilities of the property editor.

15. The apparatus of claim 13, wherein the one or more methods invoked by the editor include one or more PropertyEditor Interface methods.

16. The apparatus of claim 14, wherein if the one or more abilities include a text editing ability, the graphical user interface includes a text field entry area.

17. The apparatus of claim 16, wherein if the one or more abilities include a text editing ability, the graphical user interface further includes an entry error indicator.

18. The apparatus of claim 17, wherein the entry error indicator is only visible when an entry in the text field entry area is invalid.

19. The apparatus of claim 14, wherein if the one or more abilities include an ability to edit a property using tags, the graphical user interface includes at least one of a popup choice selection area virtual button and a current selection display field.

20. The apparatus of claim 19, wherein if the popup choice selection area virtual button is selected, a choice selection area popup is presented.

21. The apparatus of claim 14, wherein if the one or more abilities includes an ability to edit the property using a custom editor interface, the graphical user interface includes a popup custom component area virtual button.

22. The apparatus of claim 21, wherein if the one or more abilities includes an ability to edit the property using a custom editor interface, the graphical user interface further includes at least one of a text entry field and an entry error indicator.

23. The apparatus of claim 22, wherein the entry error indicator is only displayed when an invalid entry is entered in the text field entry area.

24. The apparatus of claim 21, further comprising means for presenting a custom component area in response to selection of the popup custom component area virtual button, wherein the custom component area includes a custom editor for the property.

25. A computer program product in a computer readable medium for editing a property, comprising:

first instructions for identifying one or more methods invoked by a property editor associated with the property;

second instructions for selecting a graphical user interface based on the one or more methods invoked by the property editor; and

third instructions for providing the graphical user interface for use in editing the property.

26. The computer program product of claim 25, wherein the first instructions for identifying one or more methods invoked by the property editor includes instructions for identifying one or more abilities of the property editor.

27. The computer program product of claim 25, wherein the one or more methods invoked by the editor include one or more PropertyEditor Interface methods.

28. The computer program product of claim 26, wherein if the one or more abilities include a text editing ability, the graphical user interface includes a text field entry area.

29. The computer program product of claim 28, wherein if the one or more abilities include a text editing ability, the graphical user interface further includes an entry error indicator.

30. The computer program product of claim 29, wherein the entry error indicator is only visible when an entry in the text field entry area is invalid.

31. The computer program product of claim 26, wherein if the one or more abilities include an ability to edit a property using tags, the graphical user interface includes at least one of a popup choice selection area virtual button and a current selection display field.

32. The computer program product of claim 31, wherein if the popup choice selection area virtual button is selected, a choice selection area popup is presented.

33. The computer program product of claim 26, wherein if the one or more abilities includes an ability to edit the property using a custom editor interface, the graphical user interface includes a popup custom component area virtual button.

34. The computer program product of claim 33, wherein if the one or more abilities includes an ability to edit the property using a custom editor interface, the graphical user interface further includes at least one of a text entry field and an entry error indicator.

35. The computer program product of claim 34, wherein the entry error indicator is only displayed when an invalid entry is entered in the text field entry area.

36. The computer program product of claim 33, further comprising fourth instructions for presenting a custom component area in response to selection of the popup custom component area virtual button, wherein the custom component area includes a custom editor for the property.

37. A method of editing a property, comprising:

identifying one or more methods invoked by a property editor for the property, wherein the one or more methods invoked by the editor include one or more PropertyEditor Interface methods, and wherein the property is a data type;

selecting a graphical user interface based on the one or more methods invoked by the property editor;

providing the graphical user interface for use in editing the property.

39. The method of claim 37, wherein if the one or more methods includes at least one of a getAsText method and a setAsText method, the graphical user interface includes a text field entry area and an entry error indicator.

40. The method of claim 37, wherein if the one or more methods include a getTags method, the graphical user interface includes a popup choice selection area virtual button and a current selection display field.

41. The method of claim 37, wherein if the one or more methods includes at least one of a supportsCustomEditor method and a getCustomEditor method, the graphical user interface includes a popup custom component area virtual button.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application: **Hartel et al.**	§	
Serial No.: **09/882,172**	§	Group Art Unit: **2174**
Filed: **June 14, 2001**	§	Examiner: **Ke, Peng**
For: **Property Editor Graphical User Interface Apparatus, Method and Computer Program Product**	§	Attorney Docket No.: **AUS920010225US1**

REQUEST FOR REINSTATEMENT OF APPEAL

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

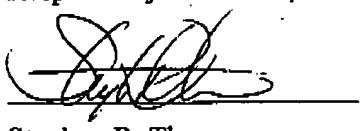
A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. A one-month extension of time is believed to be necessary. Please charge the fee for the one-month extension to Deposit Account No. 50-3157. No additional extension of time is believed to be necessary. If, however, an additional extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to Deposit Account No. 50-3157.

REMARKS

Applicants hereby request reinstatement of appeal. A Supplemental Appeal Brief is submitted herewith addressing all of the issues and rejections raised in the Final Office Action issued April 8, 2005. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: August 5, 2005

Respectfully submitted,



Stephen R. Tkacs
Reg. No. 46,430
Agent for Applicants

Duke W. Yee
Reg. No. 34,285
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants